BEST AVAILABLE IMAGES

Defective images within this document are accurate representation of The original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-200351

(43)Date of publication of application : 04.08.1995

(51)Int.Cl. G06F 11/28

(21)Application number : 06-000233

(22)Date of filing : 06.01.1994

(71)Applicant : HITACHI LTD
HITACHI TOHOKU SOFTWARE KK

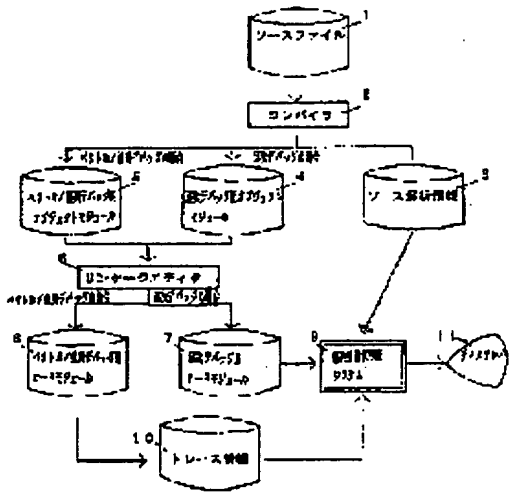
(72)Inventor : WAGAMITSU TOMOYUKI
SASAKI SHOGO
NUNOHIRO EIJI
YAMADA MASANORI

(54) PROGRAM DEBUG METHOD AND PROGRAM DEBUG SUPPORT DEVICE

(57)Abstract:

PURPOSE: To efficiently debug a program using a matrix element by comparing a threshold level set during program debugging with the selected matrix element to provide an output of the matrix elements not contained within a threshold level range thereby easily eliminating incorrect matrix element.

CONSTITUTION: When a matrix element verification system 9 is started, source analysis information 3, a sequential debugging load module 7, a source program in a source file 1 are received in the case of sequential debug, and a menu for verifying the matrix element is displayed on a display device 11. In the case of vector/parallel debug, a vector/parallel debug module 8 is executed and a menu for verifying a matrix element is displayed on the display device 11. In this case, an optional matrix element is selected and a threshold level comprising an upper limit and a lower limit is set onto the matrix element. Then the threshold level and the matrix element are compared during program debug and a matrix element not contained within a range of the threshold level is displayed or printed out.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平7－200351

(43)公開日　平成７年(1995)８月４日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　11/28	３１５　Ａ	9290－5B		

審査請求　未請求　請求項の数４　ＯＬ　（全７頁）

(21)出願番号　特願平6－233

(22)出願日　平成６年(1994)１月６日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(71)出願人　000233538
日立東北ソフトウェア株式会社
宮城県仙台市青葉区一番町２丁目４番１号

(72)発明者　我満　智幸
宮城県仙台市青葉区一番町二丁目４番１号
日立東北ソフトウエア株式会社内

(72)発明者　佐々木　祥吾
宮城県仙台市青葉区一番町二丁目４番１号
日立東北ソフトウエア株式会社内

(74)代理人　弁理士　秋田　収喜

最終頁に続く

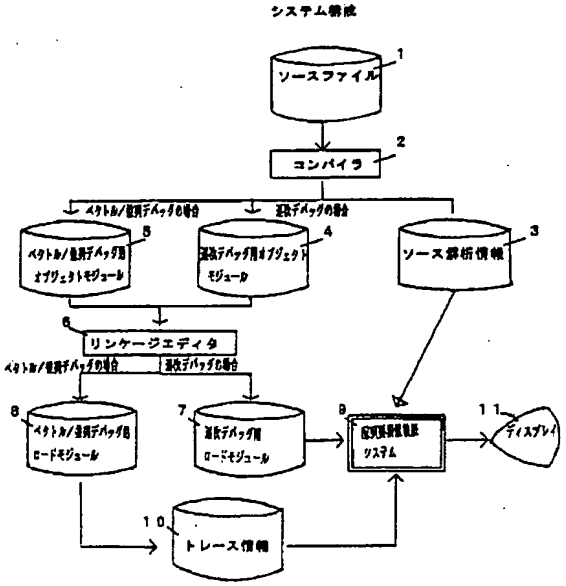
(54)【発明の名称】　プログラムデバッグ方法およびプログラムデバッグ支援装置

(57)【要約】

【目的】　不正な配列要素を容易に知り得、配列要素を使用するプログラムを効率良くデバッグすること。

【構成】　デバッグ対象のプログラムが使用する任意の配列要素を選択し、さらにその配列要素の上限値および下限値から成る閾値を設定し、プログラムのデバッグ中に前記設定された閾値と前記選択した配列要素の値とを比較し、閾値の範囲に収まらない配列要素を出力する。

図１



1

【特許請求の範囲】

【請求項１】　配列要素を使用するプログラムデバッグ方法において、デバッグ対象のプログラムが使用する任意の配列要素を選択し、その配列要素の上限値および下限値から成る閾値を設定する閾値設定手段と、前記設定された閾値と前記選択した配列要素の値とを比較する比較手段とを設け、前記プログラムのデバッグ中に前記設定された閾値と前記選択した配列要素の値とを比較し、閾値の範囲に収まらない配列要素を出力することを特徴とするプログラムデバッグ方法。

【請求項２】　前記閾値は、負、ゼロ、正の値を含む任意の値であることを特徴とする請求項１記載のプログラムデバッグ方法。

【請求項３】　多次元の配列要素については、所望の次元を前記閾値設定手段によって指定し、閾値との比較結果は指定された次元の配列を２次元のマップ形式で出力することを特徴とする請求項１記載のプログラムデバッグ方法。

【請求項４】　デバッグ対象のプログラムが使用する任意の配列要素を選択し、その配列要素の上限値および下限値から成る閾値を設定する閾値設定手段と、前記プログラムのデバッグ中に前記設定された閾値と前記選択した配列要素の値とを比較し、閾値の範囲に収まらない配列要素を抽出する比較手段と、この比較手段で抽出された閾値の範囲に収まらない配列要素を出力する出力手段とを備えたことを特徴とするプログラムデバッグ支援装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、プログラムデバッグ方法およびその支援装置に係り、特に構造解析や流体力学など配列要素を使用するプログラムをデバッグする際のプログラムデバッグ方法およびその支援装置に関するものである。

【０００２】

【従来の技術】構造解析や流体力学など配列要素を使用するプログラムをデバッグする際に、不正な配列要素の値が存在すると、計算結果が不正になったり、異常終了してしまう。

【０００３】そこで、不正な配列要素の値が存在したかどうかを調べるために、デバッグ中に使用した配列要素をプログラム言語の出力文によって出力し、あるいはデバッガによって表示出力し、利用者に調査させる方法が実施されていた。

【０００４】しかし、全ての配列要素を調査するといった作業は負担が大きく、効率も悪いという問題があった。

【０００５】そこで、配列要素の値を動的に参照できるデバッガが開発されている。

【０００６】

2

【発明が解決しようとする課題】しかし、単に配列要素の値を２次元のマップ形式で出力するのみであるため、配列要素の値が不正であるかどうかは、依然として利用者自身で調査しなければならない。このため、デバッグ効率が依然として悪いという問題がある。

【０００７】本発明の目的は、不正な配列要素を容易に知り得、配列要素を使用するプログラムを効率良くデバッグすることができるプログラムデバッグ方法および支援装置を提供することである。

【０００８】

【課題を解決するための手段】上記目的を達成するために本発明は、基本的には、デバッグ対象のプログラムが使用する任意の配列要素を選択し、その配列要素の上限値および下限値から成る閾値を設定する閾値設定手段と、設定された閾値と前記選択した配列要素の値とを比較する比較手段とを設け、前記プログラムのデバッグ中に前記設定された閾値と前記選択した配列要素の値とを比較し、閾値の範囲に収まらない配列要素を表示出力または印字出力するようにしたことを特徴とする。

【０００９】この場合、閾値は、負、ゼロ、正の値を含む任意の値を設定するようにし、また多次元の配列要素については、所望の次元を指定し、閾値との比較結果は指定された次元の配列を２次元のマップ形式で表示するようにした。

【００１０】また、本発明のプログラムデバッグ支援装置は、デバッグ対象のプログラムが使用する任意の配列要素を選択し、その配列要素の上限値および下限値から成る閾値を設定する閾値設定手段と、前記プログラムのデバッグ中に前記設定された閾値と前記選択した配列要素の値とを比較し、閾値の範囲に収まらない配列要素を抽出する比較手段と、この比較手段で抽出された閾値の範囲に収まらない配列要素を出力する出力手段とを具備させた。

【００１１】

【作用】上記手段によれば、プログラムのデバッグに際し、任意の配列要素を選択したうえ、その配列要素の上限値および下限値から成る閾値を設定しておく。すると、プログラムのデバッグ中に、閾値と配列要素の値とが比較され、閾値の範囲に収まらない配列要素が表示出力または印字出力される。

【００１２】これにより、配列要素の値をいちいち調べることなく、不正な配列要素を極めて容易に知ることができ、配列要素を使用するプログラムを効率良くデバッグすることができ。

【００１３】また、多次元の配列要素に関しては、閾値との比較結果が２次元のマップ形式で表示される。これにより、多次元の配列要素を使用するプログラムに関しても不正な配列要素を極めて容易に知ることができる。

【００１４】

【実施例】以下、本発明を図示する実施例に基づき詳細

に説明する。

【００１５】図１は、本発明を適用した配列要素値検証システムの一実施例を示すシステム構成図であり、デバッグ支援装置としての配列要素値検証システム９には、ソースファイル１に格納されたデバッグ多種のソースプログラムをコンパイラ２でコンパイルした時のソース解析情報３、逐次デバッグ用ロードモジュール７、ベクトル／並列デバッグ用ロードモジュール８のトレース情報１０が入力され、検証結果はディスプレイ１１に出力されるようになっている。

【００１６】ここで、コンパイラ２はソースファイル１に格納された図２に示すようなＦＯＲＴＲＡＮソースプログラム１００の字句解釈、および構文解析を行い、図３に示すようなソース解析情報３を生成し、また逐次デバッグまたはベクトル／並列デバッグ用コードを生成するため、コンパイラ２に対する指示文により逐次デバッグ用オブジェクトモジュール４またはベクトル／並列デバッグ用オブジェクトモジュール５を生成する。

【００１７】ここで、ソース解析情報３はソースプログラム１００中の変数Ｉ，配列Ａ，…，…Ｘの名称１２やメモリ中の配列の領域の位置を示す相対アドレス１３およびその他の情報から構成されている。

【００１８】逐次デバッグ用オブジェクトモジュール４またはベクトル／並列デバッグ用オブジェクトモジュール５はリンケージエディタ６によって連結編集され、図４に示すような変数・配列領域７０を有する逐次デバッグ用ロードモジュール７またはベクトル／並列デバッグ用ロードモジュール８として生成される。

【００１９】そして、配列要素値検証システム９を起動すると、逐次デバッグの場合、ソース解析情報３と逐次デバッグ用ロードモジュール７およびソースファイル中のソースプログラムを入力し、ディスプレイ１１に配列要素の検証のための画面を表示する。また、ベクトル／並列デバッグの場合、ベクトル／並列デバッグ用ロードモジュール８を実行し、図５に示すように配列名管理テーブル１４を有するトレース情報１０を取得し、次にソース解析情報３とトレース情報１０およびソースファイル中のソースプログラムを入力し、ディスプレイ１１に配列要素の検証のための画面を表示する。

【００２０】利用者は、ディスプレイ１１の配列要素のマップ形式表示によって配列要素値の検証を行う。

【００２１】ここで、トレース情報１０はベクトル／並列デバッグ用ロードモジュール８を実行させた際にファイル中に出力されるもので、このトレース情報１０中の配列名管理テーブル１４は、配列名１６、ファイルの要素値アドレス１７から構成され、要素値アドレス１７によって配列名１６で示される配列の要素値１５を取り出せるようになっている。

【００２２】次に、図６を参照して逐次デバッグの際の配列値検証処理の流れについて説明する。この場合、図

２のソースプログラム１００をデバッグした結果、配列Ｘの結果が不正であったと仮定する。

【００２３】まず、利用者は、配列要素値検証システム９の起動に際し、逐次デバッグを行うか、ベクトル／並列デバッグを行うかのデバッグ方法の環境設定を行う。なお、環境設定はデバッグ方法を変更する場合についてのみ行う。

【００２４】環境設定の後、配列要素値検証システム９を起動すると、該システム９はソースファイル中のソースプログラム１００を取り込んだ後、図７に示すような配列選択画面７０をディスプレイ１１に表示し、その配列選択画面７０中のソース表示エリアにソースプログラム１００を表示する。

【００２５】そこで、利用者はスクロールバー１８によってデバッグしたい配列がある位置まで表示内容を更新し、結果不正となった配列Ｘを発見したならば、その配列Ｘをドラッグすることにより、検証対象の配列Ｘを選択する。この選択操作によって配列Ｘは図７のように網かけ状態になる。

【００２６】検証対象の配列Ｘの選択が終了したならば、システム９は図８に示す検証範囲設定画面８０をディスプレイ１１に表示した後、配列Ｘに関する配列名、型、長さ、次元数、宣言値等の情報をソース解析情報から抽出し、図示のように表示する。

【００２７】この状態で、利用者は、２次元表示するためのＸ軸、Ｙ軸の情報２０、検証する配列要素の範囲（次元）２１を設定する。

【００２８】次に、利用者は、検証対象の配列Ｘの上限値と下限値とから成る閾値を設定すべく、閾値設定メニュー２２を選択操作した後、図示しないキーボードから閾値を入力する。この閾値としては、正、ゼロ、負の任意の値を入力することができる。

【００２９】そこで、検証動作を開始させると、システム９はリンケージエディタ６によって逐次デバッグ用ロードモジュール７を生成させ、この生成された逐次デバッグ用ロードモジュール７を実行する（ステップ１０１）。

【００３０】そして、その実行中にソース解析情報３から配列要素の先頭アドレスと相対アドレス１３によってメモリ中から各配列要素の値または変数を順次取得し（ステップ１０２～１０４）、その配列要素の中で利用者が選択した配列Ｘが出現したならば、その配列Ｘの値と利用者が設定した閾値とを比較する（ステップ１０５）。この比較の結果、配列Ｘの値が閾値を構成する上限値と下限値の間に収まっているかどうかを図９の検証画面９０にマップ形式で表示する（ステップ１０６）。

【００３１】図９においては、利用者が選択した検証対象の配列名２３、Ｘ軸およびＹ軸の次元２４、閾値２５が上部に表示され、その下部に、配列Ｘの値が２次元マップ形式で表示され、閾値２５の範囲内に収まっている

5

配列値、閾値２５以下の配列値、閾値２５以上の配列値が白黒表示の場合は３種類の濃淡あるいは網かけ形式のインジケータ２７でそれぞれ表示される。この場合、インジケータ２７で示されるのは、配列値そのものではなく、配列の位置である。これによって、利用者が選択した配列要素のどの位置の値が意図していない不正な値になっているかを容易に知ることができ、配列Ｘが結果不正となる原因を対話形式で追及して行くことができる。

【００３２】なお、カラー表示の場合は、３種類の色によって区別されて表示される。また、ｎ次元配列の場合は、その次元を示す添字２６を入力することにより、対応する次元の配列を表示させることができる。

【００３３】また、検証画面９０の下部の拡大メニュー２８または縮小メニュー２９を選択することによってマップ形式の配列値の表示サイズを自由に拡大または縮小して表示させることができる。

【００３４】また、中断メニュー３０を選択するか、マップ形式の配列値のインジケータ２７の点灯表示が終了した時に、マップ表示の格子を選択すると、配列要素値を参照することができる。

【００３５】一方、閾値２５は、図９（ｂ）に示すように、正の場合は「０」を白丸とし、正の方向を太線で表示し、「０」の場合は黒丸で表示し、また「－2.0～1.0」のように詳細指定された場合は「－2.0～1.0」の範囲を太線で表示するようになっている。

【００３６】次に、ベクトル／並列デバッグの場合について図１０のフローチャートに基づいて説明する。

【００３７】この場合は、逐次デバッグの場合と同様に、検証対象の配列を選択し、さらに閾値を設定する。なお、ベクトル／並列デバッグの場合は複数の配列を選択することができる。従って、図８の検証範囲設定画面８０では、希望する配列のそれぞれについて選択する。

【００３８】この状態でシステム９を起動すると、システム９はリンケージエディタ６にベクトル／並列デバッグ用ロードモジュール８を作成させる（ステップ２０１）。

【００３９】そして、そのロードモジュール８を実行し、トレース情報１０を得る（ステップ２０２）。

【００４０】配列要素のトレース情報は、ループを抜けた直後に配列の要素値１５としてトレース情報ファイルに格納される。

【００４１】次にシステム９は、利用者が選択した配列の個数分だけ配列要素値を取り出し、その配列要素値とトレース情報１０と比較し（ステップ２０３～２０６）、図９に示したように、それぞれの配列値が閾値を構成する上限値と下限値の間に収まっているかどうかを

6

検証画面９０にマップ形式で表示する（ステップ２０７）。

【００４２】なお、図９においては、閾値の範囲内に含まれる配列要素も識別可能にしているが、閾値の範囲内に含まれない配列要素のみを表示するようにしてもよい。

【００４３】また、検証結果はディスプレイに表示しているが、プリンタに印字出力するようにしてもよい。

【００４４】

【発明の効果】以上説明したように、本発明は、デバッグ対象のプログラムが使用する任意の配列要素を選択し、さらにその配列要素の上限値および下限値から成る閾値を設定し、プログラムのデバッグ中に前記設定された閾値と前記選択した配列要素の値とを比較し、閾値の範囲に収まらない配列要素を出力するようにしたので、配列要素の値をいちいち調べることなく、不正な配列要素を容易に知り得、結果不正あるいは異常終了となる原因を追及し、その結果、配列要素を使用するプログラムを効率良くデバッグすることができる。

【００４５】また、多次元の配列要素に関しては、閾値との比較結果が２次元のマップ形式で表示される。これにより、多次元の配列要素を使用するプログラムに関しても不正な配列要素を極めて容易に知ることができる。

【図面の簡単な説明】

【図１】本発明の一実施例を示すシステム構成図である。

【図２】配列要素を有するデバッグ対象のプログラムの一例を示す説明図である。

【図３】ソース解析情報の一例を示す説明図である。

【図４】逐次デバッグ用ロードモジュールの中の変数・配列領域の一例を示す説明図である。

【図５】トレース情報の一例を示す説明図である。

【図６】逐次デバッグを行う時の処理の流れを示すフローチャートである。

【図７】配列選択画面の一例を示す説明図である。

【図８】検証範囲設定画面の一例を示す説明図である。

【図９】比較検証結果を表示する検証画面の一例を示す説明図である。

【図１０】ベクトル／並列デバッグを行う時の処理の流れを示すフローチャートである。

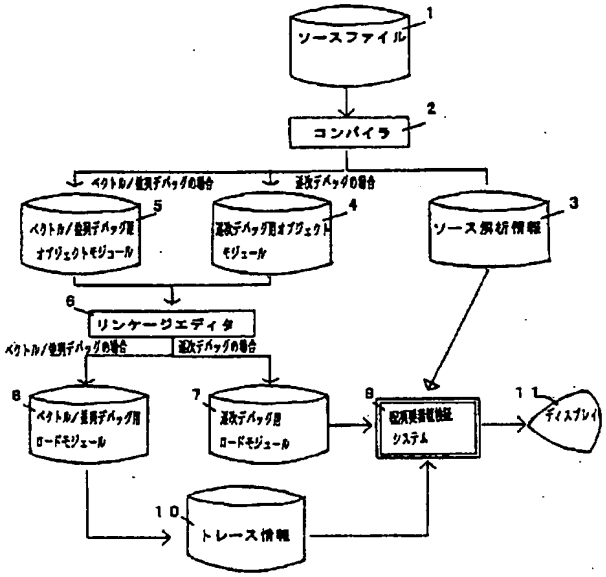
【符号の説明】

１…ソースファイル、２…コンパイラ、３…ソース解析情報、７…逐次デバッグ用ロードモジュール、８…ベクトル／並列デバッグ用ロードモジュール、９…配列要素値検証システム、１０…トレース情報、７０…配列選択画面、８０…検証範囲設定画面、９０…検証画面。

【図１】

図１

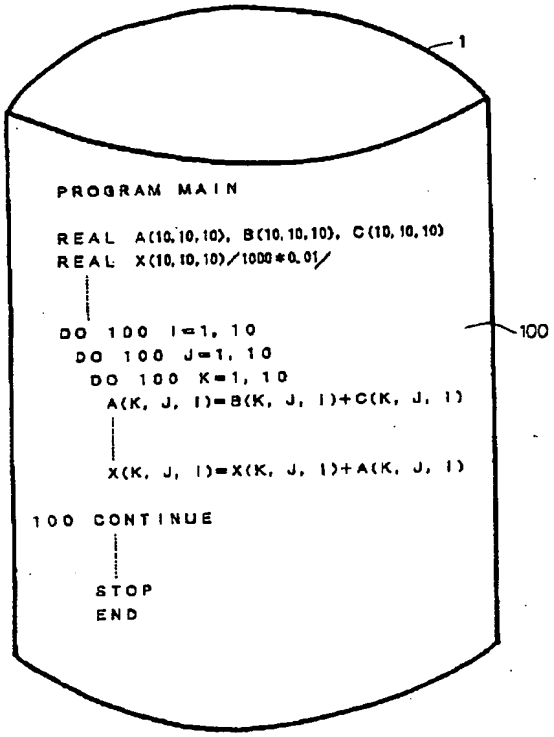
システム構成



【図２】

図２

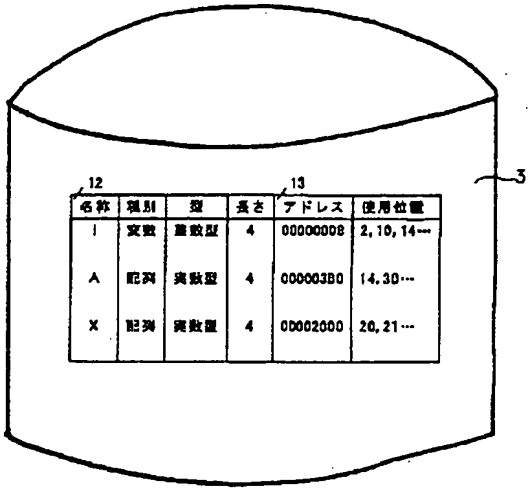
ソースプログラム例



【図３】

図３

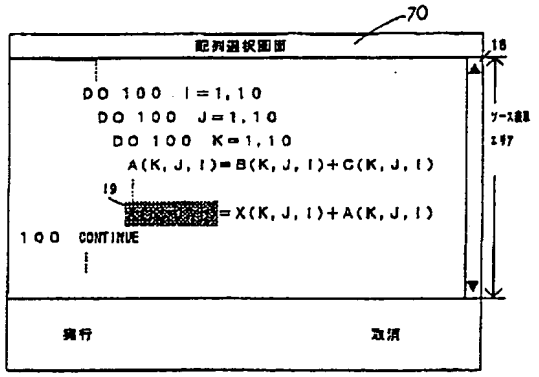
ソース解析情報



【図７】

図７

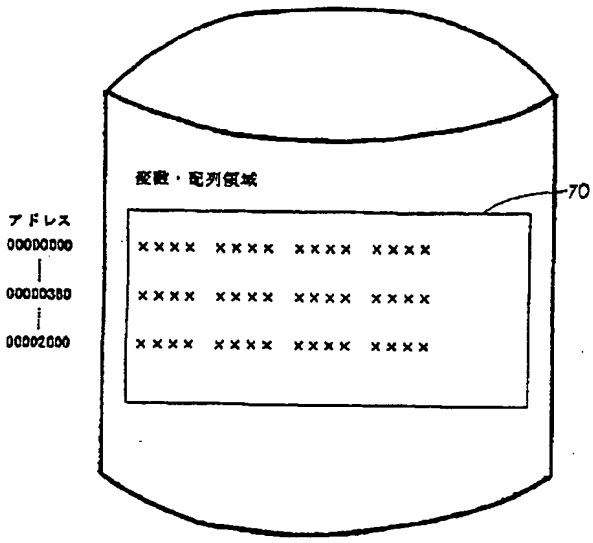
配列選択画面



【図４】

図４

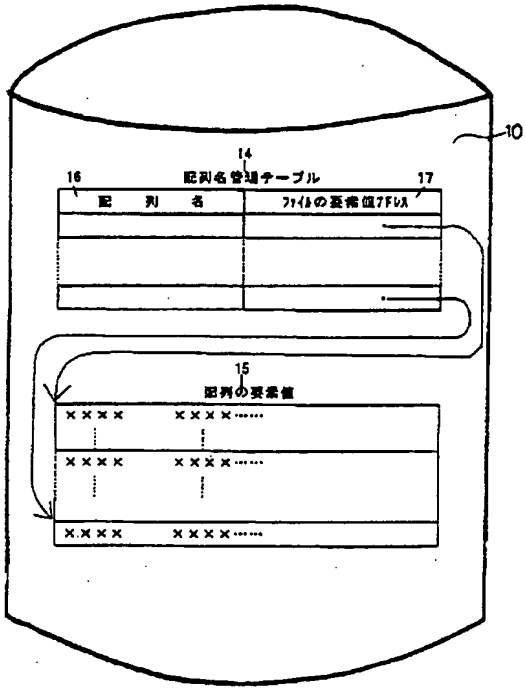
遅次デバッグ用ロードモジュール



【図５】

図５

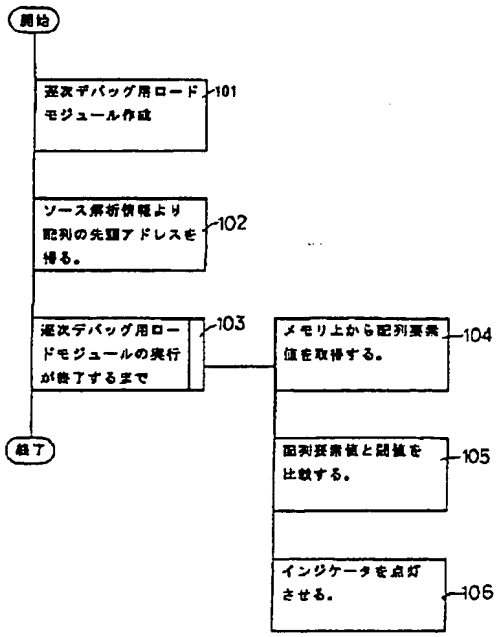
トレース情報



【図６】

図６

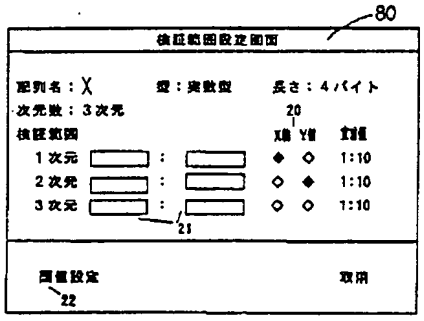
処理の流れ(遅次デバッグ)



【図８】

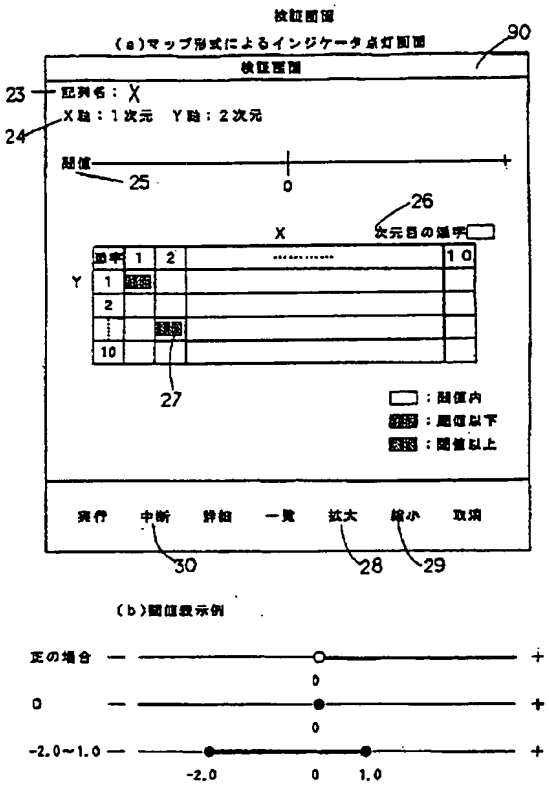
図８

検証範囲設定画面



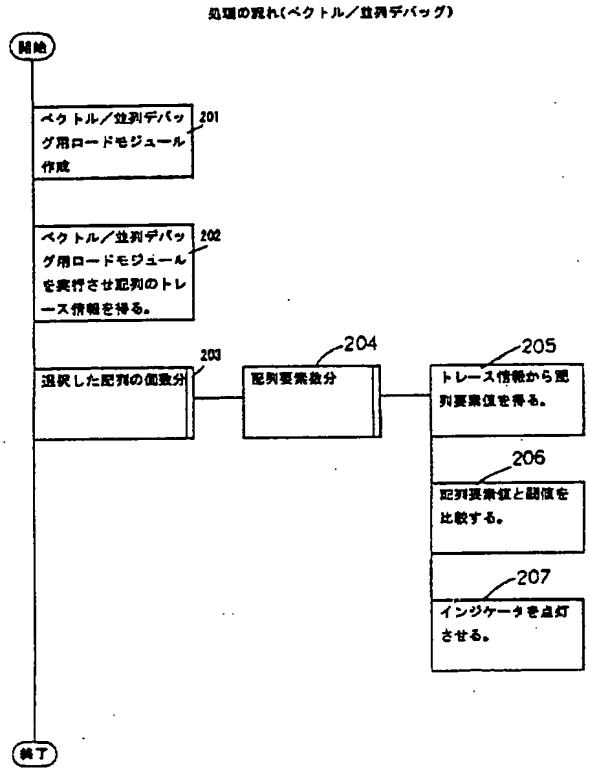
【図9】

図9



【図10】

図10



フロントページの続き

(72)発明者　布広　永示
神奈川県横浜市戸塚区戸塚町5030番地　株式会社日立製作所ソフトウェア開発本部内

(72)発明者　山田　雅則
神奈川県横浜市戸塚区戸塚町5030番地　株式会社日立製作所ソフトウェア開発本部内